IBM Pascal directory

MICROTEX 666

Still keying in programs? Forget it! This program is available for telesoftware downloading on Microtex 666 (page *6663#.)

Turbo Pascal lacks the ability to directly load the DOS's directory from within an executing program. The listing below includes a Pascal routine (LOADDIRECTORY) and a demonstration program to print out a directory listing on an IBM PC or compatible.

(Unfortunately, we've misplaced the name of the author of this routine. Anyone recognising this as their own work should call our Sydney office to arrange payment — Ed)

```
{ Program to demonstrate use of loaddirectory procedure by printing
out a directory listing.  Use of wildcards and the perimiter string
is supported.                                                       }

type anystring = string[32];
     direntry = record
                  attribute:byte;
                  time      :integer;
                  date      :integer;
                  sizelsw   :integer;
                  sizehigh  :integer;
                  name      :array[1..13] of byte;
                end;
var directory             :array[1..256] of direntry;
    temp,filecount,count :integer;
    filespec              :anystring;
    size,ts               :real;

procedure loaddirectory(filespec:anystring);
{ this procedure loads all directory entries matching the filespec into
the array defined above.  filecount in set to the number of matches found}
var dtaseg,dtaofs,destseg,destofs,temp,count:integer;
    registers:record
                ax,bx,cx,dx,bp,si,di,ds,es,flags:integer;
              end;  { see turbo pascal manual for details on how to    }
    error:boolean;  { call msdos functions.see dos technical   }
                    { reference manual for details of each routine called }

begin
    with registers do
    begin
        ax:=$2f shl 8;
        msdos(registers);
        dtaseg:=es;
        dtaofs:=bx;
    end;
    with registers do { get firest matching filenames of the disk }
    begin
        ax:=$4e shl 8;
        ds:=seg(filespec); { get segment and offset addresses of name }
        dx:=ofs(filespec)+1;{ add one to go past length of byte }
        cx:=0;
        mem[ds:dx+length(filespec)]:=0; {terminate name with a null }
        msdos(registers);
    end;

    filecount:=0;
    if not odd(registers.flags) then {if carry set there has been an error }
    begin                            { so dont continue }
        repeat
            filecount:=filecount+1;
ddress)     destseg:=seg(directory[filecount]); { get destination segment & a
            destofs:=ofs(directory[filecount]);
 are )      count:=20;                          { skip the first 20 byte, they
            repeat                              { reserved for msdos }
              count:=count+1;
              temp:=mem[dtaseg:dtaofs+count];  {code entry into buffer }
              mem[destseg:destofs+count-21]:=temp; { then blank out rest of }
            until (count>30) and (temp=0);       { our buffer }
            for count:=count to count+(42-count) do
              mem[destseg:destofs+count-21]:=0;
            registers.ax:=$4f shl 8;
            msdos(registers);                    { get next match }
            if odd(registers.flags) then         { check for error }
            error:=true
            else error:=false;
        until error;                             { error means there are no }
    end                                          { more matches }
  else write('File not found ');
     { this error is printed if the first call generated an error }
  end;

function convertdate(dt:integer):anystring;      { convert files date }
var temp:integer;
    st,st2:anystring;
begin
      str(dt mod 32,st2);
      if length(st2)=1 then st2:='0'+st2; { put in leading zeros }
      st:=st2+'/';                        { and slashes }
      temp:=lo(dt) div 32;
      if odd(hi(dt)) then temp:=temp+8;   { refer to tech. man. for more }
      str(temp,st2);                      { details of storage format }
      if length(st2)=1 then st2:='0'+st2;
      st:=st+st2+'/';
      str(hi(dt) div 2+1980,st2);
      if length(st2)=1 then st2:='0'+st2;
      st:=st+st2;
      convertdate:=st;
end;

begin
     if paramcount=0 then filespec:='*.*' else filespec:=paramstr(1);
     { if there is a parameter get it else use the wildcard }

if pos('.',filespec)=0 then filespec:=filespec+'.*';
{ if there is no extention then add a wildcard extention }

loaddirectory(filespec); {load in all matching entries }
for filecount:=1 to filecount do { and print them out}
   begin
      for count:=1 to 13 do write(chr(directory[filecount].name[count]));
      write('      ',convertdate(directory[filecount].date));
      ts:=directory[filecount].sizelow;
      if ts<0 then
         ts:=ts+65536.0;{ts}                     { convert to unsigned}
      size:=65536.0*directory[filecount].sizehigh+ts;
      { the size is 32 bits so we must put the result in the real }
      write(size:8:0, { justify and exclude decimal places }
      writeln;
   end;
end.
```

VZ Frog

by A Alley

Frog begins with a brief instruction screen and asks for the difficulty level (1 to 5). The program then draws a scene of the swamp with the full moon, several water plants and a large frog. Unfortunately, this frog is suffering from a permanent energy crisis. You, as the player, must try to keep him alive by making him eat as many of the insects flying around as possible. This requires a good deal of energy, and so too many misses will result in the frog's untimely demise. The insects get smarter as the game proceeds, and tend to duck out of the way just before the frog eats them.

```
10 CLS:PRINT:PRINTTAB(10)"[redacted]"
20 PRINTTAB(14)"[BY]":PRINTTAB(9)"[ANDREW ALLEY]":PRINT
30 PRINT" CATCH THE BUGS FOR POINTS AND"
40 PRINT"  ENERGY. USE <RETURN>,<[]>AND"
50 PRINT"  <SPACE> TO CONTROL THE FROG.":PRINT
60 PRINT"   PRESS ANY KEY TO CONTINUE":FORT=1TO10:I$=INKEY$:NEXT
70 IFINKEY$="",70
100 CLEAR400:DIMA$(2,5),B(2),B1(2),C(2),C1(2),H$(7),HS(7)
110 FORT=0TO7:H$(T)="????????????":HS(T)=250:NEXT
200 DATA"[redacted]","[redacted]"
210 DATA"[redacted]","[redacted]"
220 DATA"[redacted]","[redacted]"
230 DATA"[redacted]","[redacted]"
240 DATA"[redacted]","[redacted]"
250 DATA"[redacted]","[redacted]"
260 FORT=0TO5:FORU=0TO2:READA$(U,T):NEXTU,T
270 TI=200:RN=6:SC=0:FORT=0TO2:B(T)=28672:C(T)=RND(12)+10:NEXT
275 CLS:PRINT:INPUT"DIFFICULTY 1-5";DF:IFDF<1ORDF>5,275
277 DF=(10-(DF*2))+20
280 POKE30776,255
310 FORT=28672TO29151:POKET,128:NEXT:COLOR5
320 PRINT@26,"[redacted]";
325 PRINT@58,"[redacted]";
330 PRINT@92,"[redacted]";
335 PRINT@127,"[redacted]":COLOR2
340 PRINT@417,"[redacted]";
345 PRINT@448,"[redacted]";:COLOR5
350 PRINT@349,"[redacted]";
355 PRINT@380,"[redacted]";:COLOR1
360 PRINT@413,"[redacted]";
365 PRINT@446,"[redacted]";
370 PRINT@479,"[redacted]";
390 FORT=29152TO29183:POKET,175:NEXT:COLOR1
392 PRINT@32,USING"####0";SC
395 FORT=0TO5:PRINT@T*32+259,A$(0,T);:NEXT
397 FORT=1TO10:I$=INKEY$:NEXT:IFSC>50,450
400 IFINKEY$=CHR$(13),800
410 IFINKEY$=":",900
420 IFINKEY$=" ",950
430 TI=TI-.5:PRINT@0,USING"#####";TI;
440 IFTI<=0,2000
450 FORT=0TO2:POKEB(T)+C(T),128
460 IFB(T)<28864,B1(T)=32:GOTO510
470 IFB(T)>29056,B1(T)=-32:GOTO510
480 IFC(T)<10,C1(T)=1:GOTO510
490 IFC(T)>26,C1(T)=-1:GOTO510
500 IFRND(INT(RN))=1,B1(T)=(RND(3)-2)*32:C1(T)=RND(3)-2
510 B(T)=B(T)+B1(T):C(T)=C(T)+C1(T)
520 POKEB(T)+C(T),120+RND(8)*16:NEXT:POKE28671,1:POKE28671,2
530 GOTO400
800 FORT=0TO5:PRINT@T*32+259,A$(1,T);:NEXT
810 FORT=0TO5:PRINT@T*32+259,A$(2,T);:NEXT
820 S=28967:C=1:GOSUB1000
830 FORT=0TO5:PRINT@T*32+259,A$(1,T);:NEXT
840 GOTO395
900 FORT=0TO5:PRINT@T*32+259,A$(1,T);:NEXT
```

PROGRAM FILE

```
910 S=28999:C=1:GOSUB1000
920 GOTO395
950 S=29031:C=1:GOSUB1000
960 GOTO395
1000 S=S+1:C=C+1:IFPEEK(S)<>128,GOTO1100
1005 POKES,188:TI=TI-1:PRINT@0,USING"#####";TI;
1008 IFTI=<0,2000
1009 POKE28671,1:POKE28671,2
1010 IFC<20,1000ELSEPOKES,128
1020 S=S-1:C=C-1:POKES,128:IFC<=2,RETURN
1025 POKE28671,1:POKE28671,2
1030 GOTO1020
1100 FORT=0TO2:IFS=B(T)+C(T),SC=SC+1:TI=TI+DFELSE1120
1110 B(T)=28672:C(T)=RND(12)+10:SOUND30,1;31,1
1115 PRINT@32,USING"####0";SC;:RN=RN-.025:IFRN<2,RN=2
1120 NEXT:POKES,128:GOTO1020
2000 POKES,128:S=S-1:C=C-1:IFC<=1,2005ELSE2000
2005 SOUND4,1;0,1;4,1;0,1;4,6;6,4;8,5;0,1;9,4;0,1;6,4
2010 SOUND0,1;8,4;0,1;4,6
2040 PRINT@259,"▀▀▀▀";
2050 PRINT@291,"▀  ▀";
2060 PRINT@323,"▀  ▀";
2070 PRINT@355,"▀  ▀";
2080 PRINT@387,"▀▄▄▀";
2090 PRINT@419,"▀▀▀▀";
```

```
2095 FORT=1TO2000:NEXT
2100 POKE30776,40:IFSC*10>HS(7),2200
2110 CLS:PRINT
2120 PRINTTAB(11)"HIGH SCORES"
2130 FORT=0TO7:PRINT@136+T*32,H$(T);
2140 PRINT@150+T*32,USING"#####";HS(T):NEXT
2160 PRINT@418,"PRESS ANY KEY TO PLAY AGAIN"
2170 FORT=1TO10:I$=INKEY$:NEXT
2180 IFINKEY$="",2180
2190 GOTO270
2200 CLS:PRINTTAB(11)"HIGH SCORES"
2210 FORT=6TO0STEP-1
2220 IFSC*10>HS(T),HS(T+1)=HS(T):H$(T+1)=H$(T):F=T
2225 NEXT
2230 PRINT:PRINT" PLEASE ENTER YOUR NAME ON THE"
2240 PRINTTAB(12)"SCORE BOARD"
2250 INPUTH$(F):H$(F)=LEFT$(H$(F),12):HS(F)=SC*10:GOTO2110
```

APC is interested in programs written in any of the major programming languages for all home and small business micros. When submitting programs please include a cassette or disk version of your program, brief but comprehensive documentation, and a listing on plain white paper — typed if you have no printer. Please ensure that the software itself, the documentation and the listing are all marked with your name, address, program title, machine (along with any minimum requirements) and — if possible — a daytime phone number.

Check through the previous Program listings to see the kind of programs we prefer. As a rough guide, original ideas are always welcome, as are good implementations of utilities and applications. Obviously the programs should be well-written, easy to understand, and preferably not too long (remember that other readers have to type them in).

All programs should be fully debugged and your own original, unpublished work. We prefer to receive programs with a maximum 80-column width printed in emphasised typeface. We will try to return submissions if they are accompanied by a stamped addressed envelope of the appropriate size, but please keep a copy of everything. Programs are paid for at the rate of $20 per page of published listing. Programs APC, 215 Clarence Street, Sydney 2000.